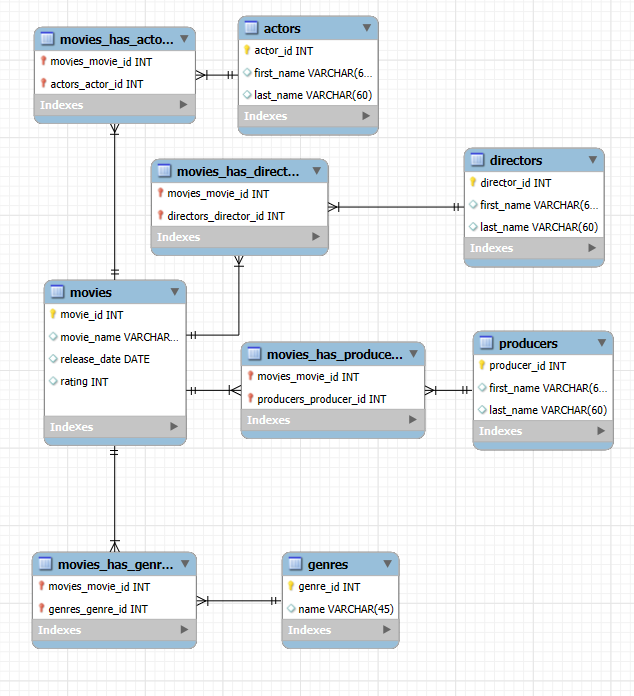
Filmdatabase



SQL Script

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema movies

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema movies

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `movies` DEFAULT CHARACTER SET utf8 ;

USE `movies` ;

-- -----------------------------------------------------

-- Table `movies`.`movies`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `movies`.`movies` (

`movie\_id` INT NOT NULL AUTO\_INCREMENT,

`movie\_name` VARCHAR(100) NULL,

`release\_date` DATE NULL,

`rating` INT NULL,

PRIMARY KEY (`movie\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `movies`.`genres`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `movies`.`genres` (

`genre\_id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NULL,

PRIMARY KEY (`genre\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `movies`.`movies\_has\_genres`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `movies`.`movies\_has\_genres` (

`movies\_movie\_id` INT NOT NULL,

`genres\_genre\_id` INT NOT NULL,

PRIMARY KEY (`movies\_movie\_id`, `genres\_genre\_id`),

INDEX `fk\_movies\_has\_genres\_genres1\_idx` (`genres\_genre\_id` ASC) VISIBLE,

INDEX `fk\_movies\_has\_genres\_movies1\_idx` (`movies\_movie\_id` ASC) VISIBLE,

CONSTRAINT `fk\_movies\_has\_genres\_movies1`

FOREIGN KEY (`movies\_movie\_id`)

REFERENCES `movies`.`movies` (`movie\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_movies\_has\_genres\_genres1`

FOREIGN KEY (`genres\_genre\_id`)

REFERENCES `movies`.`genres` (`genre\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `movies`.`actors`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `movies`.`actors` (

`actor\_id` INT NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(60) NULL,

`last\_name` VARCHAR(60) NULL,

PRIMARY KEY (`actor\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `movies`.`movies\_has\_actors`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `movies`.`movies\_has\_actors` (

`movies\_movie\_id` INT NOT NULL,

`actors\_actor\_id` INT NOT NULL,

PRIMARY KEY (`movies\_movie\_id`, `actors\_actor\_id`),

INDEX `fk\_movies\_has\_actors\_actors1\_idx` (`actors\_actor\_id` ASC) VISIBLE,

INDEX `fk\_movies\_has\_actors\_movies1\_idx` (`movies\_movie\_id` ASC) VISIBLE,

CONSTRAINT `fk\_movies\_has\_actors\_movies1`

FOREIGN KEY (`movies\_movie\_id`)

REFERENCES `movies`.`movies` (`movie\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_movies\_has\_actors\_actors1`

FOREIGN KEY (`actors\_actor\_id`)

REFERENCES `movies`.`actors` (`actor\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `movies`.`directors`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `movies`.`directors` (

`director\_id` INT NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(60) NULL,

`last\_name` VARCHAR(60) NULL,

PRIMARY KEY (`director\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `movies`.`movies\_has\_directors`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `movies`.`movies\_has\_directors` (

`movies\_movie\_id` INT NOT NULL,

`directors\_director\_id` INT NOT NULL,

PRIMARY KEY (`movies\_movie\_id`, `directors\_director\_id`),

INDEX `fk\_movies\_has\_regisseurs\_regisseurs1\_idx` (`directors\_director\_id` ASC) VISIBLE,

INDEX `fk\_movies\_has\_regisseurs\_movies1\_idx` (`movies\_movie\_id` ASC) VISIBLE,

CONSTRAINT `fk\_movies\_has\_regisseurs\_movies1`

FOREIGN KEY (`movies\_movie\_id`)

REFERENCES `movies`.`movies` (`movie\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_movies\_has\_regisseurs\_regisseurs1`

FOREIGN KEY (`directors\_director\_id`)

REFERENCES `movies`.`directors` (`director\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `movies`.`producers`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `movies`.`producers` (

`producer\_id` INT NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(60) NULL,

`last\_name` VARCHAR(60) NULL,

PRIMARY KEY (`producer\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `movies`.`movies\_has\_producers`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `movies`.`movies\_has\_producers` (

`movies\_movie\_id` INT NOT NULL,

`producers\_producer\_id` INT NOT NULL,

PRIMARY KEY (`movies\_movie\_id`, `producers\_producer\_id`),

INDEX `fk\_movies\_has\_producers\_producers1\_idx` (`producers\_producer\_id` ASC) VISIBLE,

INDEX `fk\_movies\_has\_producers\_movies1\_idx` (`movies\_movie\_id` ASC) VISIBLE,

CONSTRAINT `fk\_movies\_has\_producers\_movies1`

FOREIGN KEY (`movies\_movie\_id`)

REFERENCES `movies`.`movies` (`movie\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_movies\_has\_producers\_producers1`

FOREIGN KEY (`producers\_producer\_id`)

REFERENCES `movies`.`producers` (`producer\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

Added another table for storing soundtracks from the different movies, because why not.

Et bilde som inneholder tekst, Font, skjermbilde, line

Automatisk generert beskrivelse

Added two more columns to the *actors*, *directors* and *producers* tables. The *age* column was renamed to died\_age to show when they died. Still\_alive column removed

Adding data to tables

Here I’m adding data by writing it straight into the table. The program then auto-generates and executes the SQL query. In this situation, it’s a join table, so the code must be executed after adding data to *movies* and *genres* tables:

INSERT INTO `movies`.`movies\_has\_genres` (`movies\_movie\_id`, `genres\_genre\_id`) VALUES ('1', '1');

INSERT INTO `movies`.`movies\_has\_genres` (`movies\_movie\_id`, `genres\_genre\_id`) VALUES ('1', '2');

I normally add data with SQL queries like this:

Et bilde som inneholder tekst, skjermbilde, Font, line

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

Automatisk generert beskrivelse

A view that combines every piece of data in the database using 11 joins.

Et bilde som inneholder tekst, elektronikk, skjermbilde, programvare

Automatisk generert beskrivelse

There is also a view for the other tables, i.e. genres (above).